Model-Based Chance-Constrained Reinforcement Learning via Separated Proportional-Integral Lagrangian

Baiyu Peng, Jingliang Duan, Jianyu Chen, Shengbo Eben Li, *Senior Member, IEEE*, Genjin Xie, Congsheng Zhang, Yang Guan, Yao Mu, and Enxin Sun

***Abstract*— Safety is essential for reinforcement learning (RL) applied in the real world. Adding chance constraints (or probabilistic constraints) is a suitable way to enhance RL safety under uncertainty. Existing chance-constrained RL methods, such as the penalty methods and the Lagrangian methods, either exhibit periodic oscillations or learn an overconservative or unsafe policy. In this article, we address these shortcomings by proposing a separated proportional-integral Lagrangian (SPIL) algorithm. We first review the constrained policy optimization process from a feedback control perspective, which regards the penalty weight as the control input and the safe probability as the control output. Based on this, the penalty method is formulated as a proportional controller, and the Lagrangian method is formulated as an integral controller. We then unify them and present a proportional-integral Lagrangian method to get both their merits with an integral separation technique to limit the integral value to a reasonable range. To accelerate training, the gradient of safe probability is computed in a model-based manner. The convergence of the overall algorithm is analyzed. We demonstrate that our method can reduce the oscillations and conservatism of RL policy in a car-following simulation. To prove its practicality, we also apply our method to a real-world mobile robot navigation task, where our robot successfully avoids a moving obstacle with highly uncertain or even aggressive behaviors.**

***Index Terms*— Constrained control, neural networks, robot navigation, safe reinforcement learning (RL).**

I. Introduction

RECENT advances in deep reinforcement learning (RL) have demonstrated state-of-the-art performance in a variety of tasks, including video games [1]–[3], autonomous driving [4]–[7], and robotics [8]–[10]. However, most RL successes still remain in virtual environments or simulation platforms. For safety-critical real-world tasks, RL is not yet fully mature or ready to serve as an "off-the-shelf" solution. One of the reasons is the lack of safety constraints [11]. Generally, it is difficult to handle safety constraints for a neural network policy. Although many previous studies have discussed constraint-solving for a network policy [12]–[15], none of them are in the context of RL. Handling safety constraints still remains an open question in the deep RL community.

The safety constraints used in safe RL mainly fall into three categories: expected constraints, worst case constraints, and chance constraints. Especially, the popular constrained Markov decision process (CMDP) framework [16] is a special case of the expected constraints, which constrains the expected cumulative cost to be below a predetermined boundary. Many well-known safe RL algorithms build on this framework, including constrained policy optimization (CPO) [17], projection-based constrained policy Optimization (PCPO) [18], and reward constrained policy optimization (RCPO) [19]. However, these methods only guarantee constraint satisfaction in expectation, which is inadequate for safety-critical engineering applications. In this case, the probability of the constraint violation is about 50% (roughly speaking) [20]. The second type of constraint is the worst case constraint, which guarantees constraint satisfaction under any uncertain conditions. Nevertheless, the worst case constraint tends to be overly conservative, and it only supports systems with bounded noise [11]. The third form is the chance constraint, where the constraint holds with a predefined probability. The chance constraint clearly limits the occurring probability of the unsafe event, which is selected according to the different demands of users and the tasks. Therefore, the chance constraint is quite suitable for various real-world applications. In this article, we will focus on the chance-constrained RL problems, i.e., how to learn an optimal policy satisfying the chance constraints.

Next, we will briefly introduce the existing chance-constrained RL studies. Geibel and Wysotzki [21] use an indicator function to estimate the safe probability by sampling and add a large penalty term in the reward function if the safe probability is low. Then, the reshaped reward is optimized by an actor-critic method. Giuseppi and Pietrabissa [22] view the reward and safe probability as two objectives and propose

Manuscript received August 26, 2021; revised March 26, 2022; accepted April 28, 2022. This work was supported in part by the International Science and Technology Cooperation Program of China under Grant 2019YFE0100200, in part by NSF, China, under Grant 51575293 and Grant U20A20334, and in part by the Geekplus Technology Company Ltd. *(Corresponding author: Shengbo Eben Li.)*

Baiyu Peng, Jingliang Duan, Shengbo Eben Li, Genjin Xie, Congsheng Zhang, Yang Guan, and Enxin Sun are with the State Key Laboratory of Automotive Safety and Energy, School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China (e-mail: lishbo@tsinghua.edu.cn).

Jianyu Chen is with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China, and also with the Shanghai Qi Zhi Institute, Shanghai 200000, China.

Yao Mu is with the Department of Computer Science, University of Hong Kong, Hong Kong, China.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2022.3175595.

Digital Object Identifier 10.1109/TNNLS.2022.3175595

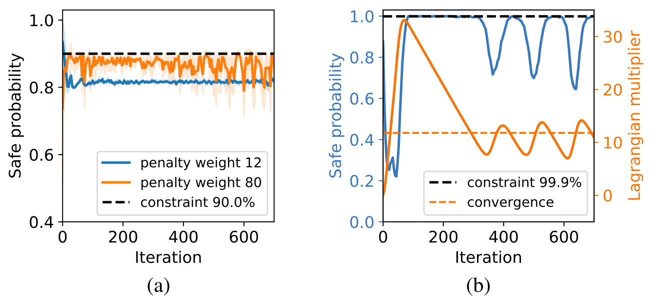


Fig. 1. Examples of learning curves for penalty and Lagrangian methods. (a) Penalty method exhibits oscillations and violates the constraint. (b) Lagrangian method exhibits Lagrange multiplier overshooting and oscillations, and further harms the policy learning.

a corresponding multiobjective RL method. Paternain *et al.* [23] derive a lower bound of the safe probability, which is employed to construct a surrogate constraint since it has an additive structure and is easier to tackle. Then, the transformed problem is solved by the Lagrangian method, which introduces a Lagrangian multiplier to balance policy performance and constraints satisfaction. To reduce the conservatism introduced by constraining a lower bound, Peng *et al.* [24] only use the lower bound to obtain an update direction but still evaluate the feasibility of the policy using original chance constraints. In addition, they employ a penalty method with increasing weight to enforce constraint satisfaction.

The previous chance-constrained RL mainly relies on the penalty method or the Lagrangian method. However, they actually face several challenges, such as poor policy performance, constraint violations, and unstable learning process [25]. The penalty methods require a well-designed penalty weight to balance the reward and constraint, which unfortunately is nontrivial and hard to tune. As shown in Fig. 1(a), a large penalty is prone to rapid oscillations and frequent constraint violations, while a small penalty always violates the constraint seriously [19]. As for the Lagrangian method, it usually suffers from the Lagrange multiplier overshooting [see Fig. 1(b)], which will lead to an overly conservative policy [23], [26]. Besides, due to the delay between the policy optimization and the Lagrange multiplier adaptation, the Lagrangian multiplier usually oscillates periodically during the training process, which further results in policy oscillations [27], [28]. In addition, most existing methods do not support model-based optimization since they all rely on a nondifferentiable indicator function to estimate the safe probability. Therefore, they can only use model-free methods to optimize the safe probability, which is generally slower than model-based methods.

To overcome the problems mentioned above, we propose a model-based separated proportional-integral Lagrangian (SPIL) method for chance-constrained RL, which can fulfill the safety requirements with a steady and fast learning process. The contributions of this article are summarized as follows.

1) This article interprets the chance-constrained policy optimization process from a feedback control perspective, which regards the penalty weight as the control input and the safe probability as the control output. Based on this, we unify the existing constraint optimization methods into the PID control methodology, in which the penalty method is formulated as a proportional controller, and the Lagrangian method is formulated as an integral controller. Then, we develop the proportional-integral Lagrangian (PIL) framework by combining the proportional and integral modules. Therefore, compared with recent chance-constrained RL studies that rely solely on the penalty or the Lagrangian methods [23], [24], [29], the proposed PIL framework gets their both merits, achieving a stable policy performance improvement without losing constraint guarantees.
2) Furthermore, to prevent the policy from overconservatism caused by multiplier overshooting, which is faced by most existing Lagrangian-based RL methods [23], [29], we draw inspiration from PID control and introduce an integral separation technique to separate the integrator out when the integral value exceeds a predetermined threshold. Then, we embed this technology into the PIL framework to propose the SPIL method for chance-constrained RL, which reaches a good tradeoff between policy performance and constraints satisfaction. The convergence of the SPIL is analyzed under some assumptions.
3) Most chance-constrained RL methods choose to estimate the policy gradient of the safe probability using model-free techniques since it is hard to identify the analytic form of the safe probability using the model knowledge [22], [23], [29]. To accelerate the training process in a model-based way, we adopt an approximated model-based gradient of the safe probability to participate in the policy optimization. The approximated gradient is proven to approach the true gradient under mild conditions. Results show that, compared with the method based on PPO, which is one of the state-of-the-art model-free RL methods, the learning speed is improved by at least five times. Finally, simulations and real-world robotics experiments demonstrate that our SPIL achieves better policy performance compared with the existing penalty and Lagrangian methods and verify its effectiveness in practical engineering problems.

This article is further organized as follows. The chance-constrained RL problem is formulated in Section II. The model-based SPIL method is proposed in Section III. In Section IV, our method is verified and compared in a car-following simulation. In Section V, the mobile robot navigation experiment is presented to prove the practicability of the algorithm. Section VI concludes this article.

II. Preliminaries

A. Problem Description

In our model-based RL setting, we assume that a dynamic model is already available, either learned by collected data or derived from prior knowledge. To indicate the gap between the model and reality, an uncertain term is included in this model. This assumption is reasonable in many physical engineering problems, such as autonomous driving, where there are many established dynamic models. Therefore, we can directly learn a policy through the given uncertainty model. To ensure the policy feasibility for real environments, a chance constraint needs to be introduced in the model-based learning process. In other words, given a reasonable uncertain term, if the policy is safe with a high probability under model uncertainty, we can usually ensure its safety in practical environments. This setting is similar to that in stochastic control [30].

The dynamic model and the chance constraint are mathematically described as

$$\begin{aligned} s_{t+1} &= f(s_t, a_t, \xi_t), \quad \xi_t \sim p(\xi_t) \\ \Pr&\left\{\bigcap_{t=1}^{N}[h(s_t) < 0]\right\} \geq 1-\delta \end{aligned} \tag{1}$$

where t is the current step, $s_t \in \mathcal{S}$ is the state, $a_t \in \mathcal{A}$ is the action, $f(\cdot,\cdot,\cdot)$ is the environmental dynamic, and $\xi_t \in \mathbb{R}^n$ is the model uncertainty following an independent and identical distribution $p(\xi_t)$. $h(\cdot)$ is the safety function defining a safe state region. Here, the chance constraint takes a joint form, which is initially brought from stochastic systems control [30]. Intuitively, it requires the probability of being safe over the finite horizon N to be at least $1-\delta$. For simplicity, we only consider one constraint, but our method can readily generalize to multiple constraints.

The chance-constrained problem is defined as maximizing an objective function J, i.e., the expected discounted cumulative reward, while keeping a high safe probability p_s

$$\begin{aligned} \max_{\pi}\ & J(\pi) = \mathbb{E}_{s_0,\xi}\left\{\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right\} \\ \text{s.t.}\ & p_s(\pi) = \Pr\left\{\bigcap_{t=1}^{N}[h(s_t) < 0]\right\} \geq 1-\delta \\ & s_{t+1} = f(s_t,a_t,\xi_t), \quad \xi_t \sim p(\xi_t) \end{aligned} \tag{2}$$

where $r(\cdot,\cdot)$ is the reward function, $\gamma \in (0,1)$ is the discount factor, and $\mathbb{E}_{s_0,\xi}(\cdot)$ is the expectation w.r.t. the initial state s_0 and uncertainty $\xi_{0:\infty}$. π is a deterministic policy, i.e., a mapping from state space $\mathcal{S}$ to action space $\mathcal{A}$. In practice, policy is usually a parameterized neural network with parameters θ, denoted as $\pi(s_t;\theta)$ or π_θ. Note that the chance constraints are defined in finite horizon, which is also widely used in stochastic model predictive control research. However, the strict recursive feasibility for the constraints is hard to establish and still remains a major challenge [30]. Instead, related articles usually assume that the recursive feasibility holds as long as the horizon is long [31], [32]. In this article, we follow their assumption and leave the strict feasibility theory in our future work.

B. Penalty and Lagrangian Methods

To find the optimal control policy for problem (2), the penalty and Lagrangian methods are widely employed in existing studies [21]–[24]. The penalty method adds a quadratic penalty term in the objective function to force the satisfaction of the constraint

$$\max_{\pi} J(\pi) - \tfrac{1}{2}K_P\left((1-\delta-p_s(\pi))^+\right)^2 \tag{3}$$

where $K_p > 0$ is the penalty weight, $p_s(\pi)$ is the joint safe probability [see (2)], and $(\cdot)^+$ means $\max(\cdot,0)$. This unconstrained problem is usually solved by gradient ascent

$$\theta^k \leftarrow \theta^{k-1} + \alpha_\theta\Big(\nabla_\theta J^{k-1} + K_P\big(1-\delta-p_s^{k-1}\big)^+\nabla_\theta p_s^{k-1}\Big) \tag{4}$$

where k means the kth iteration, $\alpha_\theta > 0$ is the learning rate, and J^k and p_s^k are short for $J(\pi_{\theta^k})$ and $p_s(\pi_{\theta^k})$, respectively. For practical applications, it is usually difficult to select an appropriate weight K_P to balance reward and constraint well. A large penalty is prone to rapid oscillations and frequent constraint violations, while a small penalty always violates the constraint seriously.

For the Lagrangian method, it first transforms the chance-constrained problem (2) into a dual problem by introduction of the Lagrange multiplier λ_I [33]

$$\max_{\lambda_I\geq 0}\min_{\pi}\mathcal{L}(\pi,\lambda_I) = -J(\pi) + \lambda_I(1-\delta-p_s(\pi)). \tag{5}$$

Then, problem (5) can be solved by dual ascent, i.e., alternatively update the Lagrange multiplier and primal variables

$$\lambda_I^k \leftarrow \big(\lambda_I^{k-1} + K_I\big(1-\delta-p_s^{k-1}\big)\big)^+ \tag{6}$$

$$\theta^k \leftarrow \theta^{k-1} + \alpha_\theta\big(\nabla_\theta J^{k-1} + \lambda_I^k\nabla_\theta p_s^{k-1}\big) \tag{7}$$

where $K_I > 0$ is the learning rate for λ_I.

As mentioned in the Introduction, the Lagrangian method usually faces the Lagrange multiplier overshooting and multiplier oscillation challenges, resulting in poor policy performance and an unstable learning process.

III. Methodology

In this section, we first reshape the penalty method and the Lagrangian method in a feedback control view, which is partly inspired by [28], [38]. Then, we unify them to formulate a proportional-integral Lagrangian method to improve the policy performance without losing the safety requirement. Finally, we introduce an integral separation technique and a model-based gradient to make the whole method practical and efficient.

A. Feedback Control View of Chance-Constrained Policy Optimization

The key insight of the proposed method comes from a deep and novel understanding of the penalty and Lagrangian methods from a control perspective. From (4) and (7), the update rule of existing methods can be expressed in a similar form

$$\theta^k \leftarrow \theta^{k-1} + \alpha_\theta\big(\nabla_\theta J^{k-1} + \lambda^k\nabla_\theta p_s^{k-1}\big) \tag{8}$$

where λ^k is actually a balancing weight. The core difference between the two methods lies in the selection of λ. For the penalty method, $\lambda^k = K_P(1-\delta-p_s^{k-1})^+$, which indicates the constraint violation at the kth iteration. For the Lagrangian method, $\lambda^k = \lambda_I^k$ in (6), which can be regarded as the sum of constraint violation over previous k iterations. This insight inspires us to review RL from a control perspective.

As shown in Fig. 2, one can view the policy optimization as a feedback control process, where θ^k is the state, λ^k is

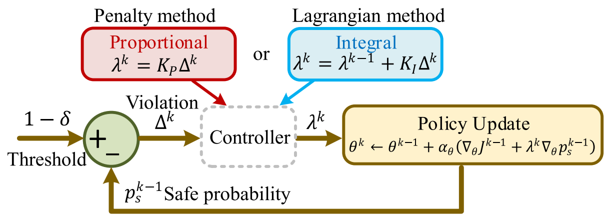

Penalty method
Proportional
$\lambda^k = K_P \Delta^k$
or
Lagrangian method
Integral
$\lambda^k = \lambda^{k-1} + K_I \Delta^k$
$1-\delta$
Threshold
Violation
Δ^k
Controller
λ^k
Policy Update
$\theta^k \leftarrow \theta^{k-1} + \alpha_\theta (\nabla_\theta J^{k-1} + \lambda^k \nabla_\theta p_s^{k-1})$
p_s^{k-1} Safe probability

Fig. 2. Feedback control view of chance-constrained policy optimization.

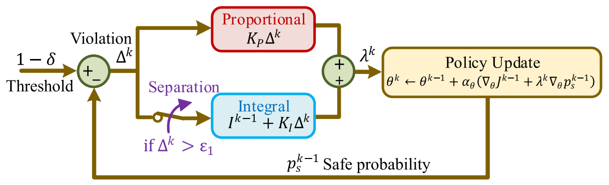

Proportional
$K_P \Delta^k$
Violation
Δ^k
$1-\delta$
Threshold
Separation
Integral
$I^{k-1} + K_I \Delta^k$
if $\Delta^k > \varepsilon_1$
λ^k
Policy Update
$\theta^k \leftarrow \theta^{k-1} + \alpha_\theta (\nabla_\theta J^{k-1} + \lambda^k \nabla_\theta p_s^{k-1})$
p_s^{k-1} Safe probability

Fig. 3. Framework of the proposed SPIL method.

the control signal, policy update (8) is the system dynamics (state transition equation), p_s^k is the system output, $1-\delta$ is the desired output, and $1-\delta-p_s^{k-1}$ is the tracking error (or constraint violation). Then, the essence of this system is the design of the controller, i.e., given the tracking error $1-\delta-p_s$, how can we decide the control signal λ? The penalty method actually adopts λ^k proportional to the violation. The Lagrangian method instead computes λ^k as the sum of previous constraint violations. In such an insight, the penalty method becomes exactly a proportional (P) controller, and the Lagrangian method becomes an integral (I) controller. Subsequently, one can easily understand the merits and flaws of these two methods by analogy. For pure "P" control, small K_P leads to steady-state error, while large K_P exhibits oscillation. This matches the phenomena that we observe in the penalty method. Similarly, the problems of overshooting and oscillation in the Lagrangian method can also be explained by properties of pure "I" control.

With this insight, we naturally propose to combine the penalty method and Lagrangian method by computing λ^k as a weighted sum of proportional and integral values, which leads to the proportional-integral Lagrangian method (PIL). The update process of PIL at $k-$iteration is given as

$$\Delta^k \leftarrow 1-\delta-p_s^{k-1} \tag{9}$$

$$I^k \leftarrow \left(I^{k-1}+\Delta^k\right)^+ \tag{10}$$

$$\lambda^k \leftarrow \left(K_P\Delta^k + K_I I^k\right)^+ \tag{11}$$

$$\theta^k \leftarrow \theta^{k-1} + \frac{\alpha_\theta}{1+\lambda^k}\left(\nabla_\theta J^{k-1} + \lambda^k \nabla_\theta p_s^{k-1}\right) \tag{12}$$

where Δ^k and I^k are proportional and integral values, respectively, with K_P and K_I as coefficients. The proportional term $K_P\Delta^k$ serves as an immediate feedback of the constraint violation. The integral term $K_I I^k$ eliminates the steady-state error at convergence. In such a framework, the penalty method and the Lagrangian method can be regarded as two special cases of PIL with $K_P > 0, K_I = 0$ and $K_P = 0, K_I > 0$, respectively. To maintain a relatively consistent step size and make policy update more stable, the gradient for θ^k in (12) is rescaled by $1/(1+\lambda^k)$.

This mechanism is expected to realize a steady learning process, just like how the proportional-integral controller works. However, there are still two key issues for practical applications as follows.

1) Integral value I easily gets overly large, leading to policy overconservatism.
2) $\nabla_\theta p_s^k$ is hard to compute, especially in a model-based paradigm.

The following subsections will explain and solve these two problems.

B. Integral Separation Technique

The integral value I^k increases according to the constraint violation Δ^k. However, when the initial policy is relatively unsafe, Δ^k can be quite large, which will cause the overshooting of I^k and λ^k. With a large λ^k in (12), the policy tends to be overly conservative since the weight of $\nabla_\theta p_s^{k-1}$ is overly large. Even worse, since the maximal safe probability is 1, the overshooting and conservatism problems can hardly recover by themselves. For example, suppose that $1-\delta = 0.999$, $p_s^k = 1$, and λ^k is already overshooting; the integral term I^k can only fall slowly with the speed of $\Delta^k = -0.001$. Therefore, the policy in such a case will deteriorate for a long time. This issue is also not well recognized and resolved in previous similar works, such as [28].

To deal with the overshooting problem of I^k and λ^k, we draw inspiration from PID control [34] and introduce an integral separation technique. As shown in Fig. 3, the integrator will be activated only when Δ^k is less than a certain value. If Δ^k is too large, the integrator will be blocked to limit the increase of I^k. Specifically, (10) is modified to

$$I^k \leftarrow \left(I^{k-1} + K_S\Delta^k\right)^+$$
$$K_S = \begin{cases} 0, & \varepsilon_1 < \Delta^k \\ \beta, & \varepsilon_2 < \Delta^k \le \varepsilon_1 \\ 1, & \Delta^k \le \varepsilon_2 \end{cases} \tag{13}$$

where K_S is the separation function, and $1 > \beta > 0$ and $\varepsilon_1 > \varepsilon_2 > 0$ are predetermined parameters. The piecewise function K_S separates the integrator out or slows it down if the error is relatively large. Such a recipe restrains the occurrence of overshooting and overconservatism. Our simulation indicates it greatly improves the performance for a large safety threshold $1-\delta$, such as $1-\delta = 99.9\%$. We refer to the combination of PIL and the separation technique as the SPIL method.

C. Model-Based Gradient of Safe Probability

Next, we will figure out how to estimate $\nabla_\theta p_s$ in (12) in an efficient way. Generally, there is not any analytic solution for a joint probability and its gradient [30], [35], i.e., they are nearly intractable. Therefore, previous researchers usually introduce an indicator function to estimate the probability through sampling [23], [24]. Due to the discontinuity of the indicator function, these methods are mostly model-free, which are generally believed to be slower than their model-based counterparts [36], [37].

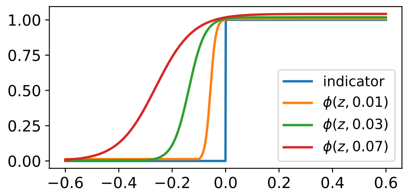


Fig. 4. Comparison of the indicator function and $\phi(z,\tau)$ with different τ's.

Inspired by recent advances in stochastic optimization [35], we introduce a model-based alternative to $\nabla_\theta p_s$, which enables us to estimate $\nabla_\theta p_s$ efficiently. We first define an indicator-like function $\phi(z,\tau)$

$$\phi(z,\tau) = \frac{1+b_1\tau}{1+b_2\tau\exp\left(-\frac{z}{\tau}\right)},$$
$$0 < b_2 < \frac{b_1}{1+b_1}, \quad 0 < \tau < 1, \tag{14}$$

where z and τ are scalar variables of the function and b_1 and b_2 are the parameters. The expected production of $\phi(z,\tau)$ over N horizon is defined as

$$\Phi(\pi,\tau) = \mathbb{E}_{s_0,\xi}\left\{\prod_{t=1}^{N}\phi(-h(s_t),\tau)\right\}. \tag{15}$$

As shown in Fig. 4, $\phi(z,\tau)$ can be intuitively regarded as a differentiable approximation of indicator function for constraint violation, and its expected product $\Phi(\pi,\tau)$ approximates joint safe probability. (To see this, one can image $\phi(z,\tau)$ in (15) as the indicator function. Then, its expected production is actually the joint safe probability.) The parameter τ controls how well the indicator function is approximated. Intriguingly, this approximation is not only an intuitive trick, and it does have strong theoretical support. Regardless of the nonlinearity and nonconvexity of $h(s_t)$, the gradient of $\Phi(\pi,\tau)$ is proven to converge to the true gradient of joint safe probability $p_s(\pi)$ as τ approaches 0 under mild assumptions [35]

$$\lim_{\tau\to 0+}\sup_{\theta\in\Theta}\nabla_\theta\Phi(\pi,\tau) = \nabla_\theta p_s(\pi) \tag{16}$$

where Θ is a small ball around the current policy network parameter. For simplicity, we omit mathematical details; interested readers are recommended to refer to [35] for a rigorous explanation.

In practice, one only needs to pick a relatively small fixed τ and compute $\nabla_\theta\Phi(\pi,\tau)$ to substitute $\nabla_\theta p_s(\pi)$, where the expectation is estimated by sampling average. Therefore, the original policy update rule of SPIL is rewritten as

$$\theta^k \leftarrow \theta^{k-1} + \frac{\alpha_\theta}{1+\lambda^k}\left(\nabla_\theta J^{k-1} + \lambda^k\nabla_\theta\Phi(\pi_{\theta^{k-1}},\tau)\right). \tag{17}$$

It should be pointed out that the introduction of $\phi(z,\tau)$ in this subsection is only used for the computation of $\nabla_\theta p_s(\pi)$. The safe probability p_s itself is still estimated through Monte Carlo sampling, i.e., suppose that there are m safe trajectories among all the M trajectories, and then, the safety probability is $p_s \approx (m/M)$.

Algorithm 1 Model-Based SPIL Algorithm

```
Initialize $\pi_{\theta^0}$, $Q_{w^0}$, $I^0 = 0$, $s_0 \in \mathcal{S}$, $k = 1$
repeat
  Rollout $M$ trajectories by $N$ steps using policy $\pi^{k-1}$
  Estimate safe probability
    $p_s^{k-1} \leftarrow \frac{m}{M}$
  Update $\lambda$ via SPIL rules
    $\Delta^k \leftarrow 1-\delta-p_s^{k-1}$
    $I^k \leftarrow (I^{k-1}+K_S\Delta^k)^+$
    $\lambda^k \leftarrow (K_P\Delta^k+K_I I^k)^+$
  Update critic:
    $\omega^k \leftarrow \omega^{k-1}+\alpha_\omega\nabla_\omega J_Q^{k-1}$
  Update actor:
    $\theta^k \leftarrow \theta^{k-1}+\frac{\alpha_\theta}{1+\lambda^k}(\nabla_\theta J^{k-1}+\lambda^k\nabla_\theta\Phi(\pi_{\theta^{k-1}},\tau))$
  $k \leftarrow k+1$
until $|Q^k-Q^{k-1}| \le \zeta$ and $|\pi^k-\pi^{k-1}| \le \zeta$
```

D. Model-Based SPIL Algorithm

Based on the aforementioned gradient, we will propose the model-based SPIL algorithm for practical applications. First, we define the state-action value of (s,a) under policy π as

$$Q^\pi(s,a) = \mathbb{E}_{\xi,\pi}\left\{\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\middle| s_0=s, a_0=a\right\}. \tag{18}$$

Thus, the expected cumulative reward J in (2) can be expressed as an N-step form

$$J(\pi) = \mathbb{E}_{s_0,\xi}\left\{\sum_{t=0}^{N-1}\gamma^t r(s_t,a_t) + \gamma^N Q^\pi(s_N,a_N)\right\}. \tag{19}$$

For large and continuous state spaces, both value function and policy are parameterized

$$Q(s,a) \cong Q(s,a;w), \quad a \cong \pi(s;\theta). \tag{20}$$

The parameterized state-action value function with parameter w is usually named the "critic," and the parameterized policy with parameter θ is named the "actor" [37].

The parameterized critic is trained by minimizing the average square error

$$J_Q^k = \mathbb{E}_{s_0,\xi}\left\{\frac{1}{2}\left(Q_{\text{target}} - Q\left(s_0,a_0;w^k\right)\right)^2\right\} \tag{21}$$

where $Q_{\text{target}} = \sum_{t=0}^{N-1}\gamma^t r(s_t,a_t) + \gamma^N Q(s_N,a_N;w^k)$ is the N-step target. The rollout length N is identical to the horizon of chance constraint.

The parameterized actor is trained via gradient ascent in (17). In particular, we first compute J and Φ through model rollout. Then, $\nabla_\theta J$ and $\nabla_\theta p_s$ are computed via backpropagation though time with the dynamic model [37]. In practice, this process can be easily finished by any autograd package. The pseudocode of the proposed algorithm is summarized in Algorithm 1.

E. Convergence Analysis

The convergence analysis and proof of the RL algorithms are generally difficult. Fortunately, the feedback system view of chance-constrained RL benefits not only the algorithm designing but also the convergence analysis. The convergence of RL algorithms is comparable to the stability of the feedback control system, i.e., one can analyze the convergence with the help of stability analyzing tools, such as the Lyapunov approach. The previous researchers have established some theories in the context of the iterative constrained optimization for neural networks, where the iterative algorithm is reshaped as a joint dynamic system and the convergence is analyzed by Lyapunov methods [28], [38]. In this article, we extend their method to chance-constrained RL and characterize the local convergence property of the proposed algorithm.

We first consider a new RL problem similar to (2) but with an equality constraint. This equality-constrained problem is easier to analyze, and the result can be extended to inequality-constrained problem conveniently

$$\max_{\theta} \; J(\theta) \quad \text{s.t. } p(\theta) = 0 \tag{22}$$

where $J(\theta)$ is the same objective function as in (2) and $p(\theta)$ represents $p_s(\pi_\theta) + 1 - \delta$. The optimal feasible solution of (22) is denoted as π^*.

We then make the following necessary assumptions when applying Algorithm 1 to problem (22).

Assumption 1: There is a small neighborhood Π around π^*, inside which the following holds.

1) $K_S = 1$, i.e., in (13), $\Delta^k \le \epsilon_2$ always holds, and there is no integral separation.
2) Matrix M is always positive definite, where matrix M is

$$M = -\nabla^2 J + \lambda \nabla^2 p + K_P \nabla p \nabla p^\top \tag{23}$$

$$M_{i,j} = \left(-\frac{\partial^2 J}{\partial\theta_i \partial\theta_j} + \lambda \frac{\partial^2 p}{\partial\theta_i \partial\theta_j} + K_P \frac{\partial p}{\partial\theta_i}\frac{\partial p}{\partial\theta_j}\right). \tag{24}$$

The first condition in Assumption 1 is reasonable since the integral separation mainly takes effect in the early stage when the safe probability is low, and it may not be triggered in the later stage when the policy is close to the optimum. The second condition in Assumption 1 actually demands the concavity of the objective function J and the convexity of safe probability p, which will be more likely to make matrix M positive definite. Detailed discussion is deferred in Remark 1.

Assumption 2: In each iteration, the estimation of the value function, safe probability, and its gradient is perfect, i.e., estimation errors and approximation errors are ignored in the following analysis.

With these two assumptions, the convergence theorem of the equality-constrained problem (22) can be derived.

Theorem 1: Under the assumption of 1 and 2, Algorithm 1 converges to a local optimal feasible solution π^* for the equality-constrained problem (22) starting from a policy inside the neighborhood Π.

Proof: We first reshape the updating rule in Algorithm 1 into continuous-time differential equations

$$\dot\theta = \alpha_\theta\left(\frac{\partial J(\theta)}{\partial\theta} - \lambda\frac{\partial p(\theta)}{\partial\theta}\right) \tag{25}$$

$$\dot\lambda = K_P \dot p(\theta) + K_I p(\theta). \tag{26}$$

Note that, according to Assumption 1, K_S always equals 1, thus not appearing in the equations anymore.

Calculating the derivative of (25) with respect to time t, one has

$$\ddot\theta_i = \alpha_\theta \sum_j \left(\frac{\partial^2 J}{\partial\theta_i\partial\theta_j} - \lambda\frac{\partial^2 p}{\partial\theta_i\partial\theta_j}\right)\dot\theta_j - \alpha_\theta\dot\lambda\frac{\partial p(\theta)}{\partial\theta_i} \tag{27}$$

where i represents the ith parameter.

By embedding (26) into (27), we have

$$\ddot\theta_i + \alpha_\theta\sum_j\left(-\frac{\partial^2 J}{\partial\theta_i\partial\theta_j} + \lambda\frac{\partial^2 p}{\partial\theta_i\partial\theta_j} + K_P\frac{\partial p}{\partial\theta_i}\frac{\partial p}{\partial\theta_j}\right)\dot\theta_j + \alpha_\theta K_I p(\theta)\frac{\partial p}{\partial\theta_i} = 0. \tag{28}$$

The above equation is the dynamics of the closed-loop feedback system, which is actually a damping-mass system with eternal force $\alpha_\theta K_I p(\theta)(\partial p/\partial\theta_i)$. Intuitively, if one can ensure that the damping is always positive, the energy of the system will constantly decrease, finally converging to a stable state.

Inspired by that, we define an energy function $v(\theta, \dot\theta)$ as

$$v(\theta,\dot\theta) := \frac{1}{2}\sum_i(\dot\theta_i)^2 + \alpha_\theta K_I\frac{1}{2}(p(\theta))^2. \tag{29}$$

The derivative of the energy function is

$$\begin{aligned}\dot v &= \sum_i \ddot\theta_i\dot\theta_i + \alpha_\theta K_I\sum_i p(\theta)\frac{\partial p}{\partial\theta_i}\dot\theta_i \\ &= -\alpha_\theta\sum_i\sum_j\dot\theta_i\left(-\frac{\partial^2 J}{\partial\theta_i\partial\theta_j} + \lambda\frac{\partial^2 p}{\partial\theta_i\partial\theta_j} + K_P\frac{\partial p}{\partial\theta_i}\frac{\partial p}{\partial\theta_j}\right)\dot\theta_j \\ &= -\alpha_\theta\dot\theta^\top M\dot\theta\end{aligned} \tag{30}$$

where the matrix M is as defined in Assumption 1 and is positive definite, which leads to

$$\dot v(\theta) = -\alpha_\theta\dot\theta^\top M\dot\theta < 0. \tag{31}$$

Because $v(\theta,\dot\theta)$ is the sum of squares, we have

$$v(\theta,\dot\theta) \ge 0. \tag{32}$$

The equality holds if and only if $p(\theta) = \dot\theta = 0$.

Equations (31) and (32) are exactly the Lyapunov stability conditions, so the system (28) is asymptotically stable inside the neighborhood Π, and the system energy approaches to 0

$$\lim_{t\to\infty} v(\theta,\dot\theta) = 0. \tag{33}$$

Furthermore, it holds that

$$\begin{aligned}&\lim_{t\to\infty}\dot\theta = 0\\ &\lim_{t\to\infty} p(\theta) = 0\\ &\lim_{t\to\infty}\frac{\partial J(\theta)}{\partial\theta} - \lambda\frac{\partial p(\theta)}{\partial\theta} = 0.\end{aligned} \tag{34}$$

Equation (34) means that Algorithm 1 converges to the local optimal feasible solution π^* for the equality-constrained problem (22). □

Through simple reformation, the above theorem can be extended to the convergence theorem for the inequality-constrained problem, i.e., the original problem (2).

Theorem 2: Under the assumption of 1 and 2, Algorithm 1 converges to a local optimal feasible solution π^* for the inequality-constrained problem (2) starting from a policy inside the neighborhood Π.

Proof: The inequality constraint is

$$p(\theta) \geq 0. \tag{35}$$

By introducing a slack variable z, it can be reformed into an equality constraint

$$g(\theta) = p(\theta) - z^2 = 0 \tag{36}$$

where $g(\theta)$ is the reformed constraint function.

If we combine θ and z as an expanded vector

$$\theta_z := \begin{bmatrix} \theta \\ z \end{bmatrix}$$

then the original inequality-constrained problem becomes an equality-constrained problem with variable θ_z. According to Theorem 1, this equality-constrained problem converges to a local optimal feasible solution $\theta_z^* = \{\theta^*, z^*\}$. This immediately means that the inequality-constrained problem (2) converges to a local optimal feasible solution θ^*. □

Remark 1: In (23), the matrix M is assumed to be positive definite to guarantee convergence. In fact, since $\nabla p \nabla p^\top$ is always positive definite, the last term $K_P \nabla p \nabla p^\top$ in (23) is positive definite if $K_P > 0$, which makes the matrix M more likely to be positive definite and, thus, improve convergence. This insight provides an underlying interpretation of why incorporating the penalty updating rules in the Lagrangian method improves convergence, which was also revealed by [38] and [28]. In addition, from (23), we also notice that the concavity of the objective function J and the convexity of the safe probability p also influence the convergence condition. If J is strictly concave and p is strictly convex, then M is always positive definite, which means that the second condition in Assumption 1 always holds in the whole policy space.

IV. Simulation Validation

A. Example Description

In this section, the proposed SPIL is verified and compared in a car-following simulation in Fig. 5, where the ego car expects to drive fast and closely with the front car to reduce wind drag [39], while keeping a minimum distance between the two cars with a high probability. Concretely, we assume that the ego car and front car follow a simple kinematics model, and the velocity of the front car is varying with uncertainty ξ.

The dynamics of the car-following example is given as

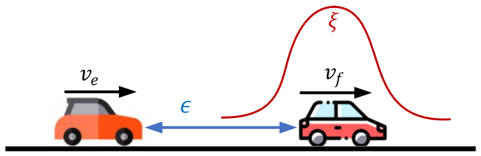


Fig. 5. Car-following scenario.

$$\begin{aligned} s_{t+1} &= As_t + Ba_t + D\xi_t \\ s &= [v_e, \quad v_f, \quad \epsilon]^\top \\ A &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -T & T & 1 \end{bmatrix} \\ B &= [T, 0, 0]^\top, \quad D = [0, T, 0]^\top \end{aligned} \tag{37}$$

where v_e (m/s) is the velocity of ego car, v_f (m/s) is the velocity of front car, and ϵ (m) is the distance between the two cars. The action $a \in (-4, 3)\text{m/s}^2$ is the acceleration of ego car. The uncertainty $\xi_t \sim \mathcal{N}(0, 0.7)$ is truncated in the interval $(-7, 7)$. $T = 0.1$ s is the simulation time step. With a chance constraint on the minimum distance, the policy optimization problem is defined as

$$\begin{aligned} \max_\pi \ & \sum_{t=0}^{\infty} \gamma^t \left(0.2v_{e,t} - 0.1\epsilon_t - 0.02a_t^2\right) \\ \text{s.t. } & \Pr\left\{\bigcap_{t=1}^{N} (\epsilon_t > 2)\right\} \geq 1 - \delta \\ & s_{t+1} = As_t + Ba_t + D\xi_t, \end{aligned} \tag{38}$$

where $v_{e,t}$ denotes the velocity of the ego car at step t.

B. Algorithm Details

Three algorithms are employed to find the nearly optimal car-following policy, including SPIL (ours), the penalty method (amounts to proportional-only PIL), and the Lagrangian method (amounts to integral-only PIL). Note that all the algorithms are trained in a model-based manner. The coefficients of SPIL are selected as $K_P = 15$ and $K_I = 0.6$ since it achieves the best results. The penalty method is sensitive to the penalty weight selection, so we adopt two weights $K_P = 12$ and 80. In fact, both of them cannot totally ensure constraint satisfaction. For the Lagrangian method, we set $K_I = 18$ because it achieved the best performance in the pretest compared to other values. The cumulative reward and safe probability in horizon N are compared under two chance constraint thresholds: 90.0% and 99.9%, i.e., $\delta = 0.1$ and $\delta = 0.001$.

Both actor and critic are approximated by fully connected neural networks. Each network has two hidden layers using rectified linear unit (ReLU) as activation functions with 64 units per layer. The optimizer for the networks is Adam [40]. The main hyperparameters are listed in Table I.

C. Results

1) Overall Performance: The learning curves of all methods under two thresholds are illustrated in Fig. 6. We emphasize

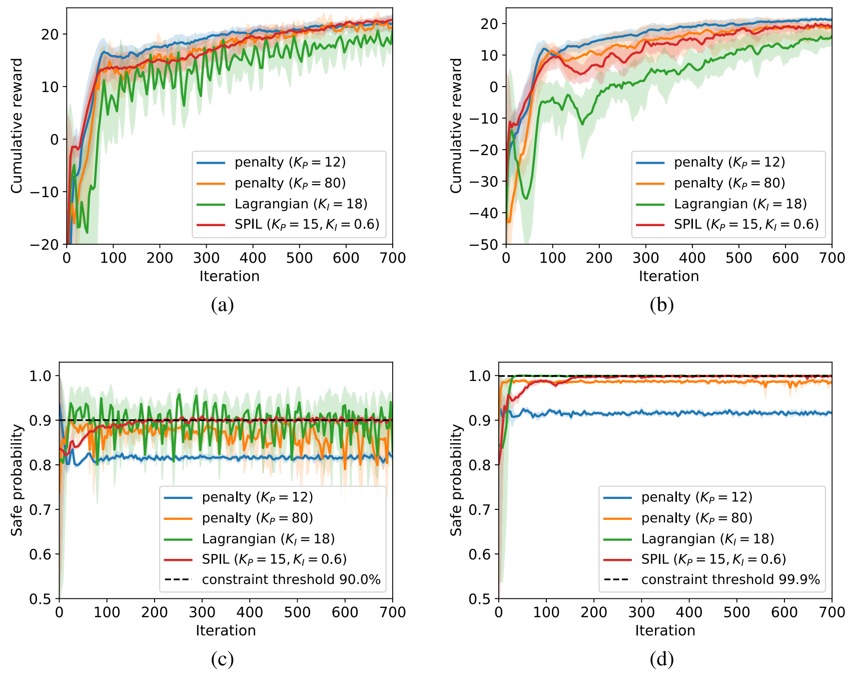


Fig. 6. Comparison of performance among SPIL, the penalty method, and the Lagrangian method. The solid lines correspond to the mean, and the shaded regions correspond to a 95% confidence interval over five runs. (a) Cumulative reward under 90.0% threshold. (b) Cumulative reward under 99.9% threshold. (c) Safe probability under 90.0% threshold. (d) Safe probability under 99.9% threshold.

TABLE I
SPIL HYPERPARAMETERS FOR SIMULATIONS

Parameters	Symbol & Value
trajectories number	$M = 4096$
constraint horizon	$N = 40$
discount factor	$\gamma = 0.99$
learning rate of policy network	$\alpha_\theta = 3e - 4$
learning rate of value network	$\alpha_\omega = 2e - 4$
parameters of K_S	$(\beta, \varepsilon_1, \varepsilon_2) = (0.3, 0.2, 0.05)$
parameters of $\phi(\cdot)$	$(\tau, b_1, b_2) = (1e - 3, 1, 0.45)$

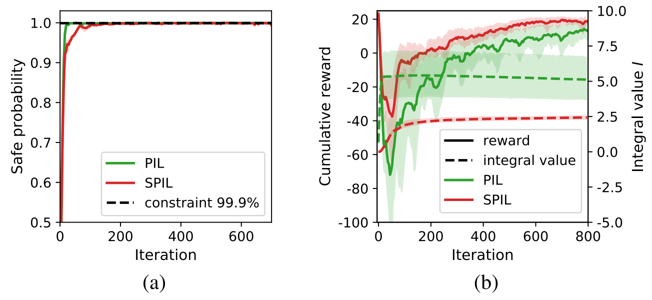


Fig. 7. Comparison of performance of SPIL with and without integral separation technique (PIL) under 99.9% threshold. The solid and dotted lines correspond to the mean, and the shaded regions correspond to 95% confidence interval over five runs. In (b), solid lines represent reward, and dotted line represent integral value. The values for SPIL are in red, and the values for PIL are in green. (a) Safe probability. (b) Cumulative reward and integral value.

that any comparison should consider both safety and reward-winning. Generally, the proposed SPIL algorithm not only succeeds to satisfy the chance constraint without periodic oscillations but also achieves the best cumulative reward among methods that meet the safety threshold.

For safety, the proposed SPIL satisfies the chance constraint in both thresholds, as shown in Fig. 6(c) and (d). While the penalty methods with two weights both fail to meet the thresholds. In addition, the large weight of $K_P = 80$ also leads to oscillation. The Lagrangian method basically satisfies the constraint but suffers from periodic oscillations under 90.0% threshold. In a feedback control view, the SPIL combines the advantages of integral and proportion control, thus having a stable learning process with no steady-state errors.

In terms of reward-wining plotted in Fig. 6(a) and (b), SPIL achieves more reward than the Lagrangian method but less than the penalty method with $K_P = 12$. This is because the penalty method actually wins high performance at the cost of constraint violation.

2) Ablation Study: To demonstrate the necessity of integral separation proposed in Section III-B, we compare the results of our method with and without integral separation with five unsafe initial policies. As shown in Fig. 7, since the initial safe probability is low, the integral value I increases rapidly at first. With a large I and λ, the policy quickly becomes 100% safe. Unfortunately, since $\Delta = 0.999 - 1 = -0.001$ is too small in (10), the decline of I and λ is quite slow, leading to an overly conservative policy with poor reward. On the contrary, once the integral separation is equipped, the above problem is immediately solved. Note that the results in Fig. 6 and Fig. 7 are not comparable since the latter are conducted under manually chosen unsafe initial policies.

The previous works typically adopt model-free methods to optimize the policy due to the discontinuity of the indicator function. In Section III-C, we propose a model-based gradient of safe probability to enable model-based optimization, which is believed to accelerate the training process. To verify this, we adopt a state-of-the-art model-free RL algorithm PPO [41] to estimate the gradient of the safe probability while keeping

TABLE II
ALGORITHM PERFORMANCE WITH DIFFERENT PARAMETERS

Value of K_P ($K_I = 0.6$, $\beta = 0.3$)	3.75	7.5	**15**	30	60
Cumulative reward	22.27 ± 0.56	21.86 ± 0.67	$\mathbf{23.01 \pm 0.96}$	21.91 ± 1.60	21.49 ± 1.61
Safe probability	$89.5 \pm 0.1\%$	$90.2 \pm 0.3\%$	$\mathbf{90.0 \pm 1.8\%}$	$91.6 \pm 1.8\%$	$88.2 \pm 6.6\%$
Value of K_I ($K_P = 15$, $\beta = 0.3$)	0.15	0.3	**0.6**	1.2	2.4
Cumulative reward	21.60 ± 0.66	22.18 ± 0.68	$\mathbf{23.01 \pm 0.96}$	21.77 ± 0.43	21.87 ± 0.50
Safe probability	$89.9 \pm 1.0\%$	$89.0 \pm 0.4\%$	$\mathbf{90.0 \pm 1.8\%}$	$90.5 \pm 0.2\%$	$89.5 \pm 0.3\%$
Value of β ($K_P = 15$, $K_I = 0.6$)	0.1	0.2	**0.3**	0.4	0.5
Cumulative reward	21.88 ± 0.90	21.25 ± 0.44	$\mathbf{23.01 \pm 0.96}$	21.86 ± 1.44	21.74 ± 0.68
Safe probability	$90.0 \pm 0.7\%$	$90.8 \pm 1.0\%$	$\mathbf{90.0 \pm 1.8\%}$	$89.9 \pm 1.1\%$	$90.1 \pm 0.4\%$

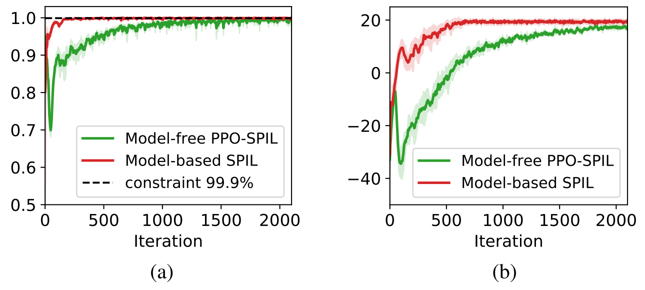


Fig. 8. Comparison of performance of SPIL with model- and PPO-based model-free optimizations under 99.9% constraint threshold. The solid lines correspond to the mean, and the shaded regions correspond to 95% confidence interval over five runs. (a) Safe probability. (b) Cumulative reward.

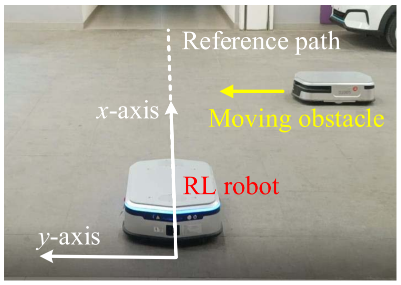


Fig. 9. Mobile robot navigation task.

the other SPIL modules the same. This baseline is denoted as model-free PPO-SPIL. The learning curves of safe probability p_s and cumulative reward J are plotted in Fig. 8, where an iteration amounts to a batch of 4096 state-action pairs for both model-based and model-free versions. Our model-based SPIL reaches the required safe probability within about 250 iterations, at least five times faster than its model-free counterpart. This improvement comes from the fact that the model-free method has first to learn an accurate cost value function before it optimizes the policy, while the model-based method makes use of the model to directly obtain a relatively precise gradient. It should be pointed out that this improvement is especially helpful for online training in the real world, where the policy should be safe as early as possible.

3) Sensitivity Analysis: To demonstrate the practicality of SPIL, we show that its high performance is relatively insensitive to hyperparameter choice. We test the algorithm across different values of K_P, K_I, and β while keeping all other parameters fixed. The results over five runs under 90.0% threshold are summarized in Table II, with the best parameters shown in bold. Even the worst case only leads to 1.8% degradation in safe probability and 6.5% degradation in cumulative reward.

V. Experimental Validation

A. Experiment Description

To demonstrate the effectiveness of the proposed method for real-world safety-critical application, we apply it to a mobile robot navigation task. As illustrated in Fig. 9, the robot aims to follow the reference path (exactly the positive x-axis) without colliding with a moving obstacle. However, it does not know the behavior or trajectory of the moving obstacle, which may be highly stochastic. Besides, the moving obstacle will not actively avoid the robot.

The robot locates its position and heading angle through a LiDAR, and it also has sensors to estimate its current velocity and angular velocity. In this experiment, we let the obstacle share its current motion information through socket communications.

B. Experiment Details

1) Dynamic Model: We first present the model used for training. The wheeled robot adopts a two-wheel differential drive architecture and takes the desired velocity v^d and desired angular velocity w^d as control commands. Nonetheless, its bottom-level control mechanism and response are unknown and varied in different conditions. Therefore, additional random terms ξ^v and ξ^ω are introduced to describe this uncertainty. To cover the stochastic behavior of obstacle, the uncertainty of the obstacle is also considered. The motions of both robot and moving obstacle are predicted through a simple kinematic model

$$s = \begin{bmatrix} P^x \\ P^y \\ \alpha \\ v \\ \omega \end{bmatrix}, \quad a = \begin{bmatrix} v^d \\ w^d \end{bmatrix}$$

$$s_{t+1} = \begin{bmatrix} P^x_t \\ P^y_t \\ \alpha_t \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ v^d_t \\ \omega^d_t \end{bmatrix} + T \begin{bmatrix} v_t \cos\alpha \\ v_t \sin\alpha \\ \omega_t \\ \xi^v_t \\ \xi^\omega_t \end{bmatrix} \tag{39}$$

where $T = 0.4$ s is the time step, P^x and P^y denote the position coordinates, α is the heading angle, v is the

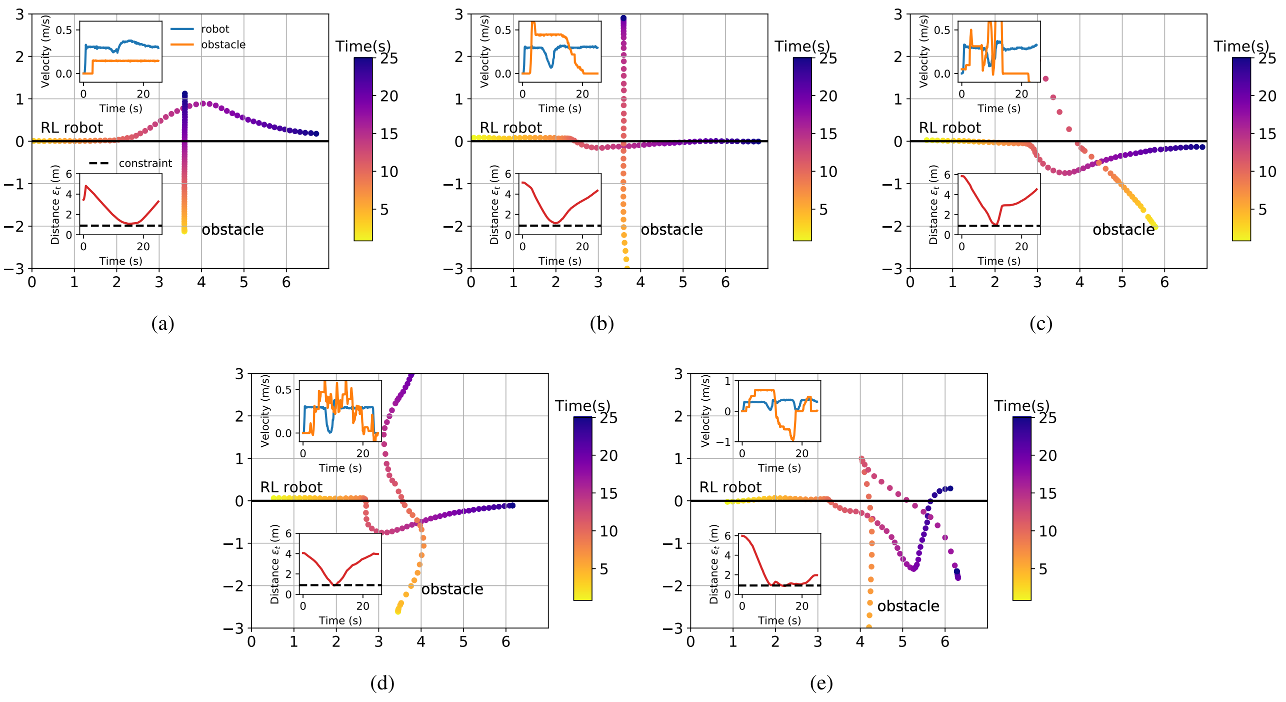


Fig. 10. Trajectories of robot and obstacle in five scenarios. All points are plotted with the same time interval, so the density of points also represents the velocity of the object. The speed of two objects and the distance between two objects are also plotted. Most of the time, our robot keeps a safe distance from the obstacle as specified in the chance constraint. (a) Scenario 1. (b) Scenario 2. (c) Scenario 3. (d) Scenario 4. (e) Scenario 5.

velocity, ω is the angular velocity, v^d is desired velocity, w^d is desired angular velocity, ξ^v is the uncertainty on the velocity, and ξ^ω is the uncertainty on the angular velocity. For the robot, the control inputs a are output by the policy network, which are also bounded by the following input constraints $|v-v^d| \leq 1.8T$ and $|\omega-\omega^d| \leq 0.8T$. The velocity and angular velocity uncertainty of the robot are set to $\xi^v \sim \mathcal{N}(0, 0.08)$ and $\xi^\omega \sim \mathcal{N}(0, 0.05)$. For the model of the obstacle, the desired velocity and angular velocity are always set as the same as its current velocity and angular velocity but with uncertainties $\xi^v \sim \mathcal{N}(0, 0.1)$ and $\xi^\omega \sim \mathcal{N}(0, 0.06)$ to indicate its stochastic behaviors. Although the true future behavior of obstacles is unknown to the ego robot, these uncertainty terms help to improve the robustness of the learned policy in real environments.

We admit the whole model is relatively naive and inaccurate, and the uncertainty term is given by experience instead of estimation. However, we find it is enough to accomplish this task. A better choice is to update the model and uncertainty online through real-world experimental data. We leave it in our further work.

2) Reward and Chance Constraints: The robot aims to follow a reference path while avoiding collisions with a moving obstacle. The start point for the robot is near (1, 0). The reference path is the positive x-axis. Therefore, the reference y-position and heading angle are both 0. The reference velocity is set to 0.3 m/s. With additional regularization on the control command, the reward of the task is defined as

$$r = -1.4\left(P^y\right)^2 - \alpha^2 - 16(v-0.3)^2 - 0.2\left(v^d\right)^2 - 0.5\left(\omega^d\right)^2. \tag{40}$$

Then, we impose the obstacle avoidance constraint. For simplicity, the robot and obstacle are both regarded as circles with a radius of 0.4 m, and the distance between their centers is denoted as ϵ. The chance constraint on the minimum distance is

$$\Pr\left\{\bigcap_{t=1}^{25}(\epsilon_t > 0.9)\right\} \geq 0.99. \tag{41}$$

TABLE III
SPIL Hyperparameters for Experiment

Parameters	Symbol & Value
trajectories number	$M = 4096$
constraint horizon	$N = 25$
learning rate of policy network	$\alpha_\theta = 3e-2$
proportional coefficient	$K_P = 60$
integral coefficient	$K_I = 0.02$
parameters of K_S	$(\beta, \varepsilon_1, \varepsilon_2) = (0.7, 0.2, 0.1)$
parameters of $\phi(\cdot)$	$(\tau, b_1, b_2) = (7e-2, 1, 0.45)$

3) Algorithmic Parameters: The network structure is exactly the same as that of the simulation. The main hyperparameters are listed in Table III.

4) Test Scenarios: The learned policy will be tested in five scenarios with different obstacle behaviors. In the former three scenarios, the obstacle behaves normally, without a sudden stop or turn. To demonstrate the robustness and high intelligence of the trained robot, the latter two scenarios are under high randomness, where the obstacle drives in a complex trajectory or even deliberately blocks the robot. We stress that, in all experiments, the robot does not know the behavior or trajectory of the obstacle in advance, and all the experiments are conducted with the same network. This

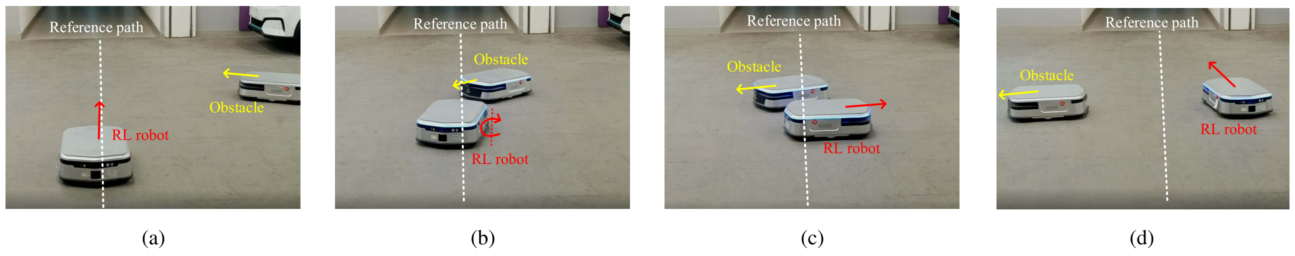


Fig. 11. Snapshots for scenario 4. (a) Obstacle initially moved toward the positive direction of the reference path. (b) Obstacle suddenly changed its direction and moved toward the robot. Thus, the robot urgently turned right to avoid it. (c) Robot passed around the obstacle from the right. (d) Robot returned to the reference path after the obstacle left. (a) Time: 10 s. (b) Time: 13 s. (c) Time: 15 s. (d) Time: 17 s.

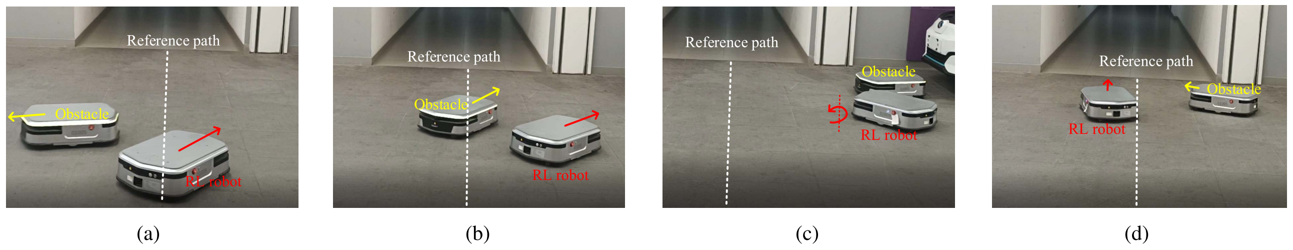


Fig. 12. Snapshots for scenario 5. (a) Robot turned right to avoid the coming obstacle. (b) Obstacle deliberately moved back and blocked the robot from returning to the reference path. Thus, the robot had to keep moving in the wrong direction. (c) Robot found an opportunity to turn left when the obstacle was not blocking. (d) Robot successfully returned to the reference path. (a) Time: 11 s. (b) Time: 15 s. (c) Time: 18 s. (d) Time: 24 s.

means that the method should have high generalization ability and intelligence to pass the five test scenarios.

C. Results

We trained the policy network using the proposed SPIL algorithm with the dynamic model and then tested the learned network on the real robot. A video for the results is available at https://youtu.be/oVDB2XqNoCU. We highly recommend readers to watch the video for an intuitive evaluation. Fig. 10 shows the control results of the learned policy in five scenarios. In Fig. 10(a), when the obstacle was moving at a low speed, the robot actively bypassed it from the left. If we increased the speed of the obstacle, the robot instead chose to stop and wait until the obstacle moved away, thereby avoiding a collision [see Fig. 10(b)]. In Fig. 10(c), the ego robot avoided the moving obstacle from the right side while tracking the reference path. These results indicate that the learned policy can adopt different strategies according to the position and speed of the obstacle to achieve collision avoidance while tracking.

In scenarios 4 [see Fig. 10(d)] and 5 [see Fig. 10(e)], the behavior of the obstacle robot was more aggressive. We controlled the obstacle to deliberately collide or block the robot to increase the difficulty of collision avoidance. See Figs. 10(d) and 11 for behavior details of the robot in scenario 4, and Figs. 10(e) and 12 for that in scenario 5. Results show that the learned policy can achieve safe movement even when the obstacle behaves aggressively, which demonstrates the superior performance of our method.

To be more specific, we list the minimum and average distances between the robot and the obstacle, as well as the safe time step ratio in Table IV. Note that the safe time step ratio denotes the portion of time steps that satisfy the constraints among all time steps in a whole run. As we declared in Section V-B2, the required minimum distance is 0.9 m, so any time step when the distance is lower than that value means a constraint violation. For the first three scenarios where the obstacle moves predictably, our method achieves a 100% safe portion. For the remaining two scenarios where the obstacle moves in a more stochastic or even aggressive trajectory, the minimum distance is lower than 0.9 m, and the safe time step ratio is relatively lower. However, the robot still successfully avoids colliding with the obstacle as shown in the videos or snapshots. This reduction in safety is reasonable and actually inevitable with the existence of the aggressive obstacle, as it deliberately moved toward the robot. Overall, these quantitative analyses demonstrate the great performance of our method under normal scenarios and decent performance under risky scenarios.

TABLE IV
ROBOT-OBSTACLE DISTANCES IN FIVE SCENARIOS

Scenarios	Minimum distance (m)	Average distance (m)	Safe time step ratio (%)
1	1.08	2.37	100.0
2	1.13	2.95	100.0
3	0.97	3.30	100.0
4	0.87	2.75	97.6
5	0.86	2.18	94.0

VI. CONCLUSION

We presented a model-based RL algorithm SPIL for chance-constrained policy optimization. Based on a feedback control view, we first reviewed and unified two existing chance-constrained RL methods to formulate a proportional-integral Lagrangian method and enhanced it with an integral separation technique to prevent policy overconservatism. To accelerate training, it also adopted a model-based gradient of safe probability for efficient policy optimization. The convergence of the algorithm was analyzed by Lyapunov methods. We demonstrated the benefits of SPIL over previous methods in a car-following simulation. To prove its practicality, it was also applied to a real-world robot navigation task, where it successfully tracked the reference path while avoiding a highly stochastic moving obstacle. In the future, we will explore the possibility of online training where the model and policy are both updated depending on the data from the real world to improve its online performance.

Acknowledgment

The authors would like to thank the Editor-in-Chief, the Associate Editor, and anonymous reviewers for the valuable comments.

References

[1] O. Vinyals *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[3] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3215–3222.

[4] J. Duan, S. E. Li, Y. Guan, Q. Sun, and B. Cheng, "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data," *IET Intell. Transp. Syst.*, vol. 14, no. 5, pp. 297–305, 2020.

[5] Y. Ren, J. Duan, S. E. Li, Y. Guan, and Q. Sun, "Improving generalization of reinforcement learning with minimax distributional soft actor-critic," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–6.

[6] J. Duan, Z. Liu, S. E. Li, Q. Sun, Z. Jia, and B. Cheng, "Adaptive dynamic programming for nonaffine nonlinear optimal control problem with state constraints," *Neurocomputing*, vol. 484, pp. 128–141, May 2022.

[7] B. Peng *et al.*, "End-to-end autonomous driving through dueling double deep Q-network," *Automot. Innov.*, vol. 4, no. 3, pp. 328–337, Aug. 2021.

[8] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–15.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[10] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, "Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 8, 2021, doi: 10.1109/TNNLS.2021.3082568.

[11] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," 2019, *arXiv:1904.12901*.

[12] S. Li, J. He, Y. Li, and M. U. Rafique, "Distributed recurrent neural networks for cooperative control of manipulators: A game-theoretic perspective," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 2, pp. 415–426, Feb. 2016.

[13] S. Li, Y. Zhang, and L. Jin, "Kinematic control of redundant manipulators using neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2243–2254, Oct. 2016.

[14] S. Li, Y. Li, and Z. Wang, "A class of finite-time dual neural networks for solving quadratic programming problems and its K-winners-take-all application," *Neural Netw.*, vol. 39, pp. 27–39, Mar. 2013.

[15] S. Li, Y. Lou, and B. Liu, "Bluetooth aided mobile phone localization: A nonlinear neural circuit approach," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 4, pp. 1–15, 2014.

[16] E. Altman, *Constrained Markov Decision Processes*, vol. 7. Boca Raton, FL, USA: CRC Press, 1999.

[17] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 22–31.

[18] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–24.

[19] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–15.

[20] P. Petsagkourakis, I. Orson Sandoval, E. Bradford, F. Galvanin, D. Zhang, and E. Antonio del Rio-Chanona, "Chance constrained policy optimization for process control and optimization," 2020, *arXiv:2008.00030*.

[21] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Intell. Res.*, vol. 24, pp. 81–108, Jul. 2005.

[22] A. Giuseppi and A. Pietrabissa, "Chance-constrained control with lexicographic deep reinforcement learning," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 755–760, Jul. 2020.

[23] S. Paternain, M. Calvo-Fullana, L. F. O. Chamon, and A. Ribeiro, "Learning safe policies via primal-dual methods," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Dec. 2019, pp. 6491–6497.

[24] B. Peng, Y. Mu, Y. Guan, S. E. Li, Y. Yin, and J. Chen, "Model-based actor-critic with chance constraint for stochastic system," in *Proc. 60th IEEE Conf. Decis. Control (CDC)*, Dec. 2021, pp. 4694–4700.

[25] M. Farina, L. Giulioni, and R. Scattolini, "Stochastic linear model predictive control with chance constraints–A review," *J. Process Control*, vol. 44, pp. 53–67, Aug. 2016.

[26] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6070–6120, 2017.

[27] B. Wah, "Improving the performance of weighted Lagrange-multiplier methods for nonlinear constrained optimization," *Inf. Sci.*, vol. 124, nos. 1–4, pp. 241–272, May 2000.

[28] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by pid Lagrangian methods," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 9133–9143.

[29] Y. Zhang, Q. Vuong, and K. Ross, "First order constrained optimization in policy space," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15338–15349.

[30] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Syst. Mag.*, vol. 36, no. 6, pp. 30–44, Dec. 2016.

[31] D. Lenz, T. Kessler, and A. Knoll, "Stochastic model predictive controller with chance constraints for comfortable and safe driving behavior of autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2015, pp. 292–297.

[32] F. Oldewurtel, C. N. Jones, and M. Morari, "A tractable approximation of chance constrained stochastic MPC based on affine disturbance feedback," in *Proc. 47th IEEE Conf. Decis. Control*, 2008, pp. 4731–4736.

[33] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[34] K. J. Astrom and L. Rundqwist, "Integrator windup and how to avoid it," in *Proc. Amer. Control Conf.*, Jun. 1989, pp. 1693–1698.

[35] A. Geletu, A. Hoffmann, and P. Li, "Analytic approximation and differentiability of joint chance constraints," *Optimization*, vol. 68, no. 10, pp. 1985–2023, Oct. 2019.

[36] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 465–472.

[37] S. E. Li, *Reinforcement Learning for Decision-making and Control*. Berlin, Germany: Springer, 2022.

[38] J. Platt and A. Barr, "Constrained differential optimization," in *Proc. Neural Inf. Process. Syst.*, 1987, pp. 612–621.

[39] F. Gao, S. E. Li, Y. Zheng, and D. Kum, "Robust control of heterogeneous vehicular platoon with uncertain dynamics and communication delay," *IET Intell. Transp. Syst.*, vol. 10, no. 7, pp. 503–513, Sep. 2016.

[40] Y. Bengio, I. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2016.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

Baiyu Peng received the B.S. degree from the School of Vehicle and Mobility, Tsinghua University, Beijing, China, in 2019, where he is currently pursuing the M.Phil. degree with the School of Vehicle and Mobility.

His current research interests include safe reinforcement learning algorithms and decision-making and control of automated vehicles.

Mr. Peng was a Finalist of the Best Student Paper Award from the IEEE Intelligent Vehicle Symposium (IV) in 2021.

Jingliang Duan received the Ph.D. degree from the School of Vehicle and Mobility, Tsinghua University, Beijing, China, in 2021.

He studied as a Visiting Student Researcher with the Department of Mechanical Engineering, University of California at Berkeley, Berkeley, CA, USA, in 2019. He is currently a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include decision and control of autonomous vehicles, reinforcement learning and adaptive dynamic programming, and driver behavior analysis.

Jianyu Chen received the bachelor's degree from Tsinghua University, Beijing, China, in 2015, and the Ph.D. degree from the University of California, Berkeley, CA, USA, in 2020.

He was working with Prof. Masayoshi Tomizuka with the University of California at Berkeley, Berkeley, CA, USA. He is currently an Assistant Professor with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University. He is working at an intersection of machine learning, robotics, and control to build intelligent systems, including autonomous vehicles and robots. His research interests include reinforcement learning, control, autonomous driving, and robotics.

Shengbo Eben Li (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 2006 and 2009, respectively.

He worked with Stanford University, Stanford, CA, USA; the University of Michigan, Ann Arbor, MI, USA; and the University of California at Berkeley, Berkeley, CA, USA. He is currently a Tenured Professor with Tsinghua University. He is the author of over 100 journals/conference papers. He is the co-inventor of over 20 Chinese patents. His active research interests include intelligent vehicles and driver assistance, reinforcement learning and distributed control, and optimal control and estimation.

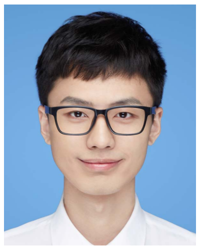
Dr. Li serves as an Associate Editor for the *IEEE Intelligent Transportation Systems Magazine* and the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

Genjin Xie received the B.S. degree from the School of Mechatronics and Vehicle Engineering, East China Jiaotong University, Nanchang, China, in 2020. He is currently pursuing the M.S. degree with the School of College of Engineering, China Agricultural University, Beijing, China.

His current research interests include decision and control of autonomous vehicles, and reinforcement learning.

Congsheng Zhang received the B.S degree from the School of Mechatronics and Vehicle Engineering, East China Jiaotong University, Nanchang, China, in 2020. He is currently pursuing the M.S. degree with the School of College of Engineering, China Agricultural University, Beijing, China.

His current research interests include constrained reinforcement learning and optimal control.



Yang Guan received the B.S. degree from the School of Mechanical Engineering, Beijing Institute of Technology, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with the School of Vehicle and Mobility, Tsinghua University, Beijing.

His research interests include decision-making of autonomous vehicles and reinforcement learning.

Mr. Guan was a recipient of the Best Student Paper Award at the IEEE International Conference on Intelligent Transportation Systems (ITSC) in 2020 and 2021.

Yao Mu received the M.Phil. degree from the School of Vehicle and Mobility, Tsinghua University, Beijing, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science, The University of Hong Kong, Hong Kong.

His research interests include reinforcement learning, representation learning, autonomous driving, and computer vision.

Enxin Sun received the B.S. degree from the College of Transportation, Shandong University of Science and Technology, Shandong, China, in 2019. He is currently pursuing the M.S. degree with the College of Engineering, China Agricultural University, Beijing, China.

His current research interests include reinforcement learning algorithms, and state estimation and control of automated vehicles.